KEYWORD SPOTTING WITH DURATION CONSTRAINED HMMS

Marcel Vasilache, Adriana Vasilache

Nokia
P.O. Box 100, 33721 Tampere, Finland
E-mail: marcel.vasilache@nokia.com

ABSTRACT

This paper focuses on a near optimal detector based on dynamic programming for voice activation. The algorithm is analysed and evaluated against the optimal detector while taking both accuracy and complexity factors into consideration. It is shown that when using the more restrictive framework of Hidden Markov models (HMMs) with state duration constraints this is improving both the accuracy of the approximate detector as well as providing a substantial boost to the detection performance.

1. INTRODUCTION

Voice activation is a desirable feature for modern voice driven user interfaces as it offers the possibility for a hands free interaction mode. In many practical use cases this can be very challenging due to reduced audio quality when speaking from a distance or due to the noisy surroundings. A typical setup for such a system requires the keyword spotter to be in a continuous listening mode for prolonged time intervals therefore placing high demands on the quality of the algorithm for which near zero false acceptance rates must be combined with very high detection rates.

Starting with methods closer to a conventional speech recognition system [1, 2, 3] detection algorithms have been extended with approaches which focus more directly on the keyword model and avoid the use of potentially problematic garbage or filler models [4, 5, 6, 7]. In the following, the paper will revise the design options for the keyword detectors and focus on the complexity and optimality aspects for an algorithm adapted from [4]. New to previous presentations, the suboptimality of this algorithm will be clearly illustrated. In the final sections a set of experiments are performed and conclusions are drawn. Concerning the lack of precision for the fast but approximate algorithm the experimental findings are rather surprising.

2. DETECTION WITH CDHMM

As in a typical speech recognition setup, the input audio waveform is transformed into a sequence of overlapping frames from which feature vectors are extracted. These observation vectors $o_{1:n} = o_1, o_2, ..., o_n$ are then modeled as a 1^{st} order Markov chain using hidden Markov models [8]. Frequently these vectors take values from a continuous space and HMMs with continuous state densities (CDHMMs) are used as the modelling tool.

In a frame synchronous detection algorithm at each frame we have to evaluate two competing hypotheses:

H_0: keyword was not spoken
H_1: keyword was spoken

When using CDHMMs one often used approach [1, 3] is to have a model for the keyword K and a filler model G for everything else (i.e. non-keyword speech, silence and environmental noise). The recognition grammar has then two entries corresponding to the two hypotheses:

H_0: $< G >$
H_1: $< G >< K >$

and leads to comparing $P(G|o_{1:n})$ and $P(GK|o_{1:n})$.

These can be expressed as

$$\frac{P(o_{1:n}|G)P(H_0)}{P(o_{1:n})} \gtrless \frac{P(o_{1:n}|GK)P(H_1)}{P(o_{1:n})}$$

which makes possible their evaluation within an HMM based framework. Here $P(H_o)$ and $P(H_1)$ are the prior probabilities for H_0 and H_1, respectively.

When using the CDHMMs in a Viterbi (i.e. best path) decoding setup we have

$$P(o_{1:n}|GK) = \max_t P(o_{1:t-1}|G)P(o_{t:n}|K)$$

with t^*, the optimal time frame for separation between the two models.

With a one state model for G [1] we can also factor $P(o_{1:n}|G)$ as

$$P(o_{1:n}|G) = P(o_{1:t^*-1}|G)P(o_{t^*:n}|G)$$

which then transforms the decision into

$$P(o_{t^*:n}|K)P(H_1) \gtrless P(o_{t^*:n}|G)P(H_0)$$

or, as a log criterion

$$\log P(o_{t^*:n}|K) - \log P(o_{t^*:n}|G) \gtrless \theta \quad (1)$$

with θ the decision threshold.

With this we can see that the filler model has two roles; providing the means for finding the optimal time segmentation t^* and giving a reference score against which model scores are evaluated. For a good detection it is therefore important that an accurate filler model be trained which can be a difficult task. In addition, in a system of several active keywords each one needs, ideally, its own filler model.

To avoid using a filler model, a criterion focusing only on the model likelihoods can be derived. Considering an uninformative filler model ($P(o_{t:n}|G,H_o) \approx P(o_{t:n}|G,H_1)$ for all t) the equation (1) becomes

$$\log P(o_{t^*:n}|K) \gtrless \xi$$

[1] or assuming this as a very close approximation in the general case

where we are still faced with the task of finding the optimal time segmentation for K. Since we must find it without the help of a filler model an acceptable method for comparing hypotheses with different frame lengths can search for the maximal average log likelihood per frame as in:

$$\max_{t \le n} \frac{\log P(o_{t:n}|K)}{n-t+1} \gtrless \delta \quad (2)$$

3. VITERBI ALGORITHM

3.1 Notations and algorithm

As mentioned in the previous section, the value $P(o_{1:n}|K)$ for the optimal state sequence is found using the Viterbi algorithm. Let us consider a CDHMM having S states with parameters (π, A, B) where A_{ik} is the log transition probability from state i to k, $B(o|s_k)$ gives the log of the state emission likelihood for all $k \in \overline{1,S}$ and π_k gives the initial log probability for state k. If we denote $P(o_{1:n}|s_k)$ the likelihood of generating the observation sequence $o_{1:n}$ while being at frame n in state s_k the Viterbi algorithm recursively computes these values for all model states as described below.

Initialization for all $k \in \overline{1,S}$.

$$\log P(o_1|s_k) = B(o_o|s_k) + \pi_k$$

Recursion $n \ge 1$ and for all $k \in \overline{1,S}$.

$$\log P(o_{1:n}|s_k) = B(o_n|s_k) + \max_i \{\log P(o_{1:n-1}|s_i) + A_{ik}\}$$

3.2 Token passing formulation

In a token passing formulation [9], each state has associated a token with the value

$$T_{s_k}(n) = \log P(o_{1:n}|s_k)$$

If we denote by s_0 a "virtual" entry state and we extend the transition matrix with a zero'th line such that $A_{0k} = \pi_k$ the algorithm can be rewritten as

Initialization for all $k \in \overline{1,S}$

$$T_{s_0}(0) = 0$$
$$T_{s_k}(0) = -\infty$$

Recursion $n \ge 1$ and for all $k \in \overline{1,S}$

$$T_{s_0}(n) = -\infty$$
$$T_{s_k}(n) = B(o_n|s_k) + \max_{i \in \overline{0,S}} \{T_{s_i}(n-1) + A_{ik}\}$$

Let us denote by $C_{\overline{0}}(A)$ the number of elements in A for which the probability is non zero hence the log is greater than $-\infty$ and including them in the maximization search makes sense. In this case if we evaluate the cost of a comparison to be similar to an addition and we leave out the costs for the B functions we can count the number of operations required by the algorithm above at $2C_{\overline{0}}(A)$ additions per frame.

In terms of memory the algorithm requires the storage of two $S+1$ sized arrays of token values for the consecutive time instants $n-1, n$. However, when the transition matrix has no cycles, as it is often the case for the speech recognition word models, one single array with a proper state updating sequence is enough.

4. NORMALIZED VITERBI ALGORITHMS

4.1 Accurate algorithms

In a direct approach, finding the optimum for equation (2) would require running in parallel $n - t_{min} + 1$ forward Viterbi algorithms where t_{min} is a minimal time instant where the search is started, assuming a maximal feasible length for the keyword K. Token values need to be stored for all the searched lengths. Even if a minimum feasible duration is considered, the memory requirements or number of computations are not reduced since state scores need to be buffered for $n - t_{max} + 1$ frames and one minimum length Viterbi search has to be ramped up each frame.

A better solution consists in buffering the state scores for $n - t_{min} + 1$ frames and running the Viterbi algorithm [2] on a time reversed model starting with the current frame n down to t_{min}.

Both cases require similar storage and computational complexity. The computation required is of $2(n - t_{min} + 1)C_{\overline{0}}(A)$ additions while $(n - t_{min} + 2)S$ values need to be stored.

4.2 Approximate algorithm

Closely following the approach at subsection 3.2, the approximate search has two values updated for each state at each frame. These are the cumulative log likelihood, as in the token passing algorithm and the length, in frames, of the summation. In this case locally optimal decisions are taken at each state for propagating the values for which the frame normalized likelihood is highest. In contrast with the previous algorithm, the entry state is left active during each frame of the search allowing the propagation of hypotheses with variable lengths for the state sequences.

Initialization for all $k \in \overline{1,S}$.

$$T_{s_0}(0) = 0 \quad L_{s_0}(0) = 0$$
$$T_{s_k}(0) = -\infty \quad L_{s_k}(0) = 0$$

Recursion $n \ge 1$ and for all $k \in \overline{1,S}$.

$$T_{s_0}(n) = 0 \quad L_{s_0}(n) = 0$$

$$i^* = \operatorname*{argmax}_{i \in \overline{0,S}} \{\frac{B(o_n|s_k) + T_{s_i}(n-1) + A_{ik}}{L_{s_i}(n-1) + 1}\}$$

$$T_{s_k}(n) = B(o_n|s_k) + T_{s_{i^*}}(n-1) + A_{i^*k}$$
$$L_{s_k}(n) = L_{s_{i^*}}(n-1) + 1$$

4.3 Computational benefits

Counting the number of operations each frame we have $C_{\overline{0}}(A)$ increments, $C_{\overline{0}}(A)$ divisions and $3C_{\overline{0}}(A) - S$ additions, assuming again equal costs for additions and comparisons. Even ignoring how we weight the costs for each operation type we can notice that this algorithm is considerably more expensive than the conventional Viterbi search but substantially less complex than the accurate methods from subsection 4.1.

[2] in a slightly modified initialization and terminal condition

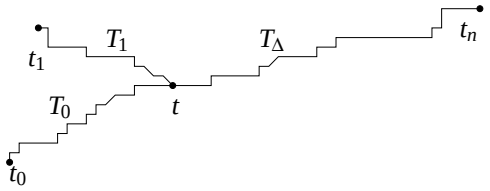

Figure 1: Local decision point

4.4 Sub-optimality and duration constrained models

Since the objective function does not satisfy the dynamic programming optimality condition the algorithm is not guaranteed to produce the optimal result. We can illustrate this with a simple drawing in figure 1 where we have decided at time t in favor of the path starting at t_0. By T_0 and T_1 we denote the token values among which we have made the selection and by T_Δ the remaining log likelihood increment until t_n. We denote the path lengths as $L_i = t - t_i + 1, i = 0,1$ and $L_\Delta = t_n - t$.

Due to the local decision at t we have $T_0/L_0 \geq T_1/L_1$ and at t_n we have the confidence equal to $(T_o + T_\Delta)/(L_o + L_\Delta)$. Was the decision at t optimal considering the time t_n ?

$$(T_o + T_\Delta)/(L_o + L_\Delta) \gtrless (T_1 + T_\Delta)/(L_1 + L_\Delta)$$

We can transform the previous comparison into

$$\frac{T_0}{L_0} + \frac{T_0}{L_0}\frac{L_\Delta}{L_1} + \frac{T_\Delta}{L_0} \gtrless \frac{T_1}{L_1} + \frac{T_1}{L_1}\frac{L_\Delta}{L_0} + \frac{T_\Delta}{L_1}$$

from which we can see that only $T_0 \geq T_1$ and $L_0 \leq L_1$ will guarantee the desired outcome. For all other values, although there is a positive bias given by the decision at t the final outcome can be reversed depending on T_Δ and L_Δ which are unknown at t.

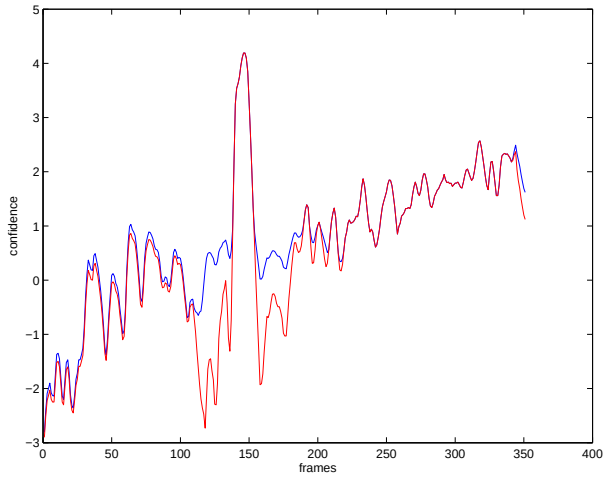

Figure 2: Confidence values for the optimal and sub-optimal algorithms

In practice it was observed that the algorithm gives most of the time quite close approximations. As example, in figure 2 a comparison is shown with the confidence levels for the accurate algorithm which are in the upper curve. For this example typical left-to-right duration unconstrained keyword models were used. It can be seen that while preserving the detection peak significant differences are located near it. This was a typical outcome for all the samples in the testing database.

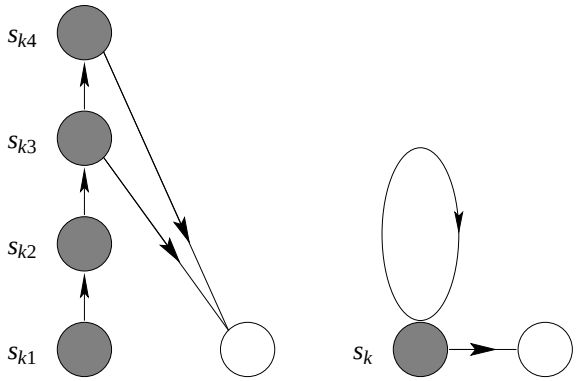

Figure 3: Duration constrained and duration unconstrained states

Although the duration unconstrained models are cheapest to implement, a more accurate modelling of the keywords should make use of hard duration constraints for each state. In figure 3 an example of such duration constrained state is shown against an unconstrained state on the right hand side. For the duration constrained case the states $s_{k1}, \ldots, s_{k4}$, share the same state emission likelihood function $B(o|s_k)$ and the duration for the state is constrained in this example at minimum 3 or maximum 4 frames.

With duration constraints the transition matrix is larger having considerable more significant transitions to evaluate. At the same time, more token values need to be stored, one for each of the states, resulting in a substantial increase in the complexity of the algorithm. However, this cost is worthwhile since these models not only produce a much closer approximation of the confidences of the accurate algorithm but are also capable of a substantially better rejection performance as will be shown in the experimental section. When these were compared against the optimal detector on the test data the difference in the confidence values were marginal in all cases. This increased accuracy comes as result of the more constrained optimization space as well as of the fact that for each state s_k there are now much more hypotheses corresponding to $s_{k1}, \ldots, s_{kDmax}$ which can survive the risk of an incorrect local decision.

5. THRESHOLD STABILITY

In an ideal case, with accurate enough models, the expressions

$$\max_{t\leq n} \frac{\log P(o_{t:n}|K,H_o)}{n-t+1}$$

and

$$\max_{t\leq n} \frac{\log P(o_{t:n}|K,H_1)}{n-t+1}$$

will result in a completely separable set of values for the testing data and the threshold δ from equation 2 will be independent on the given keyword and environment condition. In practice, the likelihood values are found to have a large variation, depending heavily on the keyword type and environment. In an attempt to stabilize such dependencies the state emission likelihoods were each frame normalized against the top scoring states in a similar approach as the on-line garbage model method from [2].

$$B_{rank}(o|s_k) = B(o|s_k) - B^Q(o)$$

where $B^Q(o)$ is the Qth percentile for the distribution of the state scores for the given model $\{B(o|s_k)|k \in \overline{1,S}\}$.

6. EXPERIMENTS

6.1 Experimental setup

A set of experiments was conducted on an in house recorded database consisting of 8 speakers and about 60 speaker specific keywords with 12 repetitions for each out of which 3 were used for training a speaker and keyword specific CDHMM.

The feature extraction is based on 13 FFT derived MEL cepstral coefficients together with their 1st order time derivatives. For noise robustness, a feature vector normalization scheme was used as in [10]. For creating the noise testing data multi-condition noise was mixed to the original data at an SNR of about 5 dB.

The training uses a simple iterative Viterbi procedure to estimate the parameters. The number of states is decided based on the keyword length while a single Gaussian density is used for the state function $B(o|s_k)$. When using duration constraints these are estimated based on the trained duration unconstrained models by doing a forced alignment on the training data and taking for each state a wide enough range to fit the alignments.

6.2 Experimental results

The results are presented in the Tables 1-2. For selected levels of the operating accuracy for the detectors the false acceptance percentage is displayed. Three test cases are shown: *ML*, ML_{dur} and $Rank_{dur}$. These correspond to using log likelihood scores with duration unconstrained models then with duration constrained ones and, finally, when using the rank based state likelihoods as presented in section 5. In the rank based case the value of the 90th percentile was used as reference.

From the tables we observe that in the noise free case all methods produce near flawless performance. However, it is obvious that the duration unconstrained models perform quite poorly for the noisy conditions. With duration constraints a substantial detection improvement is observed while with rank based scores another significant gain is seen.

Detection (%)	*100*	*98*	*95*	*90*	*80*	*70*
ML	0.03	0.02	0.01	0.01	0.01	0.00
ML_{dur}	0.06	0.03	0.03	0.02	0.01	0.01
$Rank_{dur}$	0.03	0.00	0.00	0.00	0.00	0.00

Table 1: False acceptance rates (%) for the noise free case

Detection (%)	*100*	*98*	*95*	*90*	*80*	*70*
ML	62.76	32.36	16.05	6.21	1.73	0.31
ML_{dur}	16.01	6.75	1.34	0.40	0.10	0.03
$Rank_{dur}$	15.08	3.38	0.75	0.33	0.03	0.00
Rank	42.59	16.76	10.35	4.55	0.54	0.12
Rank$*$	46.96	18.03	11.07	4.31	0.54	0.16

Table 2: False acceptance rates (%) in the noisy condition

In the noisy test case we also evaluated the effect of the approximate algorithm. For duration constrained models there were no practical differences when comparing an accurate confidence evaluation with the fast but approximate algorithm. However, for duration unconstrained models the results for the accurate evaluation in *Rank*$*$ line are in fact slightly worse than for the fast algorithm (*Rank* line). The better performance of the approximate approach ca be explained by the fact that detection peaks are preserved while in outlier situations confidences are sometimes reduced making so for a slightly better separation of the two hypotheses.

7. CONCLUSION

In this paper we have reviewed the keyword-spotting framework when using CDHMMs as detectors. The focus was on evaluating a frame synchronous setup which does not require filler models. In addition to optimal but expensive search procedures an approximate algorithm is presented and evaluated. In this context, duration constrained CDHMMs are proposed for offering a high quality of detection in the presence of noise. In addition, the quality of approximation for the confidence values is also much better. A surprising observation concerning the approximate algorithm is that although, in general, it can give significantly lower confidences, it does preserve rather closely the detection peaks. By this, when using the duration unconstrained CDHMMs the detection performance is even slightly increased in comparison with the accurate but computationally expensive detector.

REFERENCES

[1] J.G. Wilpon, L.R. Rabiner, C.-H. Lee, and E.R. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden Markov models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, no. 11, pp. 1870–1878, Nov 1990.

[2] H. Bourlard, B. D'Hoore, and J.-M. Boite, "Optimizing recognition and rejection performance in wordspotting systems," in *ICASSP'94*, Los Alamitos, CA, USA, 1994, vol. 1, pp. 373–376.

[3] K.M. Knill and S.J. Young, "Fast implementation methods for Viterbi-based word-spotting," in *ICASSP'96*, May 1996, vol. 1, pp. 522–525.

[4] J. Junkawitsch, L. Neubauer, H. Höge, and G. Ruske, "A new keyword spotting algorithm with pre-calculated optimal thresholds," in *ICSLP 96*, Philadelphia, USA, 1996, pp. 2067–2070.

[5] M.C. Silaghi and H. Bourlard, "A new keyword spotting approach based on iterative dynamic programming," in *ICASSP'2000*, Istanbul, 2000.

[6] M.C. Silaghi, "Spotting subsequences matching a HMM using the average observation probability criteria with application to keyword spotting," in *AAAI*, 2005.

[7] M.C. Silaghi and R. Vargiya, "A new evaluation criteria for keyword spotting techniques and a new algorithm," in *Interspeech/Eurospeech*, Lisboa, Portugal, 2005, pp. 1593–1596.

[8] Lawrence R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[9] S.J. Young, Russell N.H., and Thornton J.H.S, "Token passing: A simple conceptual model for connected speech recognition systems," Tech. Rep., Cambridge University Engineering Department, 1989.

[10] Viikki O., Bye D., and Laurila K., "A recursive feature vector normalization approach for robust speech recognition in noise," in *ICASSP'98*, Seattle, USA, May 1998, pp. 733–736.